NPCA 部誌
2026

灘校パソコン研究部
npca

POLARIS

目次

あいさつ

皆さんこんにちは。灘校パソコン研究部部長の akinyan と申します。本日は第 80 回灘校文化祭 ”Polaris” にご来場いただきありがとうございます。灘校パソコン研究部では、競技プログラミング[*1]や機械学習、Unity[*2]などによるゲーム作成、CTF[*3]やセキュリティ、サーバー構築など、情報系の様々な分野に日々取り組んでいます。一口にパソコンと言えども多様な「好き」を持った部員たちの集大成である展示と部誌を、ぜひ楽しんで行ってください！

この冊子について

部誌のコードやファイルは、灘校パソコン研究部の公式サイトにて公開しています。少しでもご興味を持っていただけましたら、公式サイト内の「作品」ページをご覧いただくか、「NPCA 作品」で検索してください。

執筆者の表記について

執筆者名として、本名ではなくハンドルネームが記載されています。これは、部内では本名を呼ぶことが滅多になく、ほとんどの場合部内ではハンドルネームで互いを呼び合うという慣習によるものです。また、灘校の文化に従い、学年ではなく何回生かが記されています。2026 年 5 月現在、各回生は次の学年に相当します。

- 79 回生：高校 3 年生
- 80 回生：高校 2 年生
- 81 回生：高校 1 年生
- 82 回生：中学 3 年生

[*1] 与えられた課題を解決するプログラムをいかに素早く正確に記述するかを競うプログラミングのコンテストの総称
[*2] ゲームを作るためのアプリケーション (ゲームエンジン) の一種
[*3] Capture The Flag（旗取りゲーム）の略、情報セキュリティの技術を競う競技

第 1 章

discord に LINE のメッセージを転送する

80 回生 akinyan

1.1 はじめに

こんにちは。パソコン研究部部長の akinyan です。高 2 なのですが、部誌を書くのは初めてらしいです。何故なんでしょうか。今回は discord に LINE のメッセージを転送してみたいと思います。このことをしようとした理由ですが、LINE って引継ぎするときにたまに失敗してデータが消えてしまうんですよね。なので、過去のログをいつでも見れるように discord 上に残しておきたいと思いました。

LINE の Messaging API を使って、Google Apps Script(以下 GAS と略す) に転送し、そこから discord の webhook に送って discord にメッセージが送られるみたいな感じです。

とりあえずまずはテキストのメッセージだけを discord に流すことを目標とします。

1.2 流れ

discord の webhook の URL を取得するために、転送したいチャンネルの設定を開き、連携サービスから新しいウェブフックを作成し、そのウェブフック URL をコピーしておきます。このウェブフック URL は GAS 内のコードに埋め込みます。次に LINE の公式アカウントを作り、LINE Developers でチャネルとプロバイダを作成します。そして、GAS のコードを書いたら新しいデプロイを押し、アクセスできるユーザーを全員にして、デプロイします。URL をコピーして LINE Developers の Messaging API 設定のところの Webhook 設定のところに貼り付けることで、動きます。

1.3 テキストのメッセージを discord に流す

GAS のコードをいじっていきます。

```
1 function doPost(e) {
2   const discordUrl = "DISCORD_WEBHOOK_URL";
3   const json = JSON.parse(e.postData.contents);
4   const event = json.events[0];
5
6   if (event.type === 'message' && event.message.type === 'text') {
7     const userMessage = event.message.text;
```

```
8       const payload = {
9         "content": userMessage
10      };
11
12      const options = {
13        "method": "post",
14        "contentType": "application/json",
15        "payload": JSON.stringify(payload)
16      };
17
18      UrlFetchApp.fetch(discordUrl, options);
19    }
20  }
```

このコードの重要なところをピックアップして解説をします。

2 行目で discord の webhook の URL を定義しています。
3,4 行目で LINE から送られてきた JSON を取り込んでいます。
6 行目で送られてきたものがメッセージであることを判別し、そしてそのタイプがテキストであることを判別しています。
7 行目でテキストを取り込んでいます。
18 行目で discord に向けて取り込んだテキストを送っています。
といった感じのコードが動いています。このコードをデプロイして Webhook 設定をして、LINE でメッセージを送ると次のようになりました。

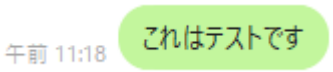


図 1.1　LINE でメッセージを送る

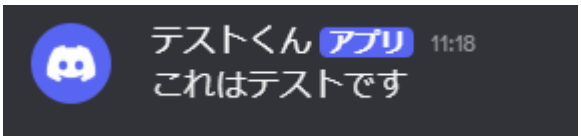


図 1.2　discord 上にメッセージが来ているのがわかる

ちゃんと discord 上にメッセージが送られていることが分かりますね。ということで第一の目標であるテキストのメッセージだけを discord に流すということは達成できたかなと思います。
しかしこの状態だと誰から送られてきたかがわかりません。ということで次の章で誰から送られてきたのかを分かるようにしましょう。

1.4 誰から送られてきたのかを判別して送る

誰から送られてきたのかを判別したいと思います。判別するためには LINE にユーザーの情報を聞かなければならないので、LINE のトークンが必要になります。発行しておきましょう。そして、GAS のコードを下のやつにします。というわけでこのコードの解説をしていきます。

```
1  function doPost(e) {
2    const discordUrl = "DISCORD_WEBHOOK_URL";
3    const lineToken = "LINE_TOKEN";
4    const json = JSON.parse(e.postData.contents);
5    const event = json.events[0];
6
7    if (event.type === 'message' && event.message.type === 'text') {
8      const userId = event.source.userId;
9      const userMessage = event.message.text;
10
11     const profileUrl = "https://api.line.me/v2/bot/profile/" + userId;
12     const response = UrlFetchApp.fetch(profileUrl, {
13       "headers": {
14         "Authorization": "Bearer " + lineToken
15       }
16     });
17     const userName = JSON.parse(response.getContentText()).displayName;
18
19     const payload = {
20       "content": `**${userName}**: ${userMessage}`
21     };
22
23     const options = {
24       "method": "post",
25       "contentType": "application/json",
26       "payload": JSON.stringify(payload)
27     };
28
29     UrlFetchApp.fetch(discordUrl, options);
30   }
31 }
```

3 行目で LINE のトークンを定義しています。

12〜16 行目で、LINE に名前を聞いています。

17 行目で名前を取り出しています。

19〜21 行目で discord に送るメッセージを **名前**: メッセージ という形式に整えています。

29 行目で discord に整えたメッセージを送っています。

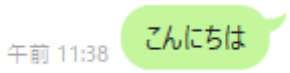


図 1.3　LINE でメッセージを送る

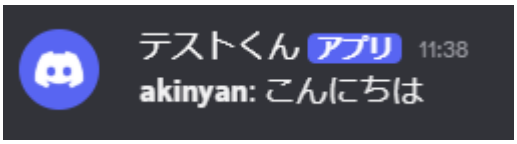


図 1.4　名前付きで来ている

1.5　おわりに

いかがでしたでしょうか。これで名前付きでテキストのメッセージなら送ることができるようになりました。しかしまだここから改善できる点があります。グループと個人チャットを区別する、画像やスタンプを送る、名前だけでなくアイコンも取得する... など、やれることはまだたくさんありますが、とりあえず本誌ではここまでとさせていただきます。

興味のある人は `https://developers.line.biz/ja/reference/messaging-api/`(公式リファレンス) を見てみると詳しいことが書いてあるので、参考になると思います。虚無の内容でありましたが、ここまで見てくださり、ありがとうございます。

第 2 章

現代の暗号化の技術について

81 回生 くま先生

2.1 はじめに

どうも、81 回生のくま先生と申します。普段は Windows のシステム構造について学んだり、ダンプファイルを生成してそれを解析したりしています。ここで、私は Windows の回復画面、その中でも「bitlocker」について興味を持ち、暗号化の技術全般についてざっくりと書こうと思いました。
ここでは身近な暗号から少し聞きなじみのないかもしれない暗号まで解説しているので、ぜひ最後まで読んでくださるとうれしいです。暗号化は単純ですが奥が深いので興味がなくとも、この記事は楽しんでもらえると思います。

警告！

途中で暗号化の脆弱性や、暗号化を行わないことのデメリットを解説する際に、攻撃についてちょびっと触れますが、この記事はその行為を推奨するものではありません。仮に許可のない攻撃を行うと刑法の「不正電磁的記録に関する罪」に該当します！絶対に実践しないでください！最悪の場合刑務所送りになりますよ！

2.2 暗号化とは？

さて、まずまずの前提として「暗号化」というものは一体何なのかということを説明していきたいと思います。
一言に「暗号化」といってもぱっと何か思い浮かぶ方はそんなにいないと思います。とはいえ、「暗号化」というものはそのままの意味で、平文（通常のデータ）を鍵またはキーと呼ばれる特定の規則性に則って暗号にする（変える）ということを指します。簡単に言ってしまえば、データをあるルールのもとで書き換える、ということです。

小学校などで友達に暗号文のようなものを送り付けた方もいるでしょう。しかしあれも、もちろんれっきとした「暗号化」の一種です。もちろん公的な暗号化の技術を使っているわけではないので、復号できるのは製作者だけだとは思いますが。

2.3 暗号化の種類

次に、暗号化の技術の種類を代表例を何個か挙げてそれらを簡単に解説したいと思います。

2.3.1 シーザー暗号

これはかなり知られている技術のうちの一つです。もちろんこの部誌を手に取っていただいて初めてご存じになる、という方もいらっしゃると思います。言ってしまえばこれは「文字ずらし」です。例えば、次の文章をご覧ください
Kxax Pzelli Cbpqfsxi
一見この文章は何の意味も持たないと思われますが、実はアルファベット順に則り 3 文字ずつずらすと（A を D にするように後ろ倒しする）と
K → N,x → a,a → d,x → a,P → S,z → c,e → h,l → o,l → o,i → l,C → F,b → e,p → s,q → t,f → i,s → v
x → a,i → l となり
Nada School Festival になりますね。
暗号界ではもっとも有名といっても過言ではない技術ですが、知ってると友達などにはマウントをとれるので、一度くらいこの暗号化を使って友達に文章を送り付けるのもいいかもしれません。

2.3.2 TLS

こちらは現代の一般的な web にアクセスするときに使われる暗号化の技術です。Transport Layer Security の略で、サーバーにある「証明書」を確認した後、公開鍵（受け取る側の鍵、要するに受け取る側のみ解読できる）により共通鍵（送る側と受け取る側に共通する鍵、要するに二人だけのヒミツ、、、）を作り、それによりデータを暗号化します。他にも途中のプロセスでハッシュ値を確認してデータや鍵が改ざんされていないか確認するシステムも組み込まれています。かなり頑丈な暗号化技術です。

2.3.3 AES

Windows の bitlocker（後ほど説明します）に使われている技術です。AES-256(256bit),AES-128(128bit) などの種類がありますが、数字に関しては強度が異なるだけで中身は同じです。ざっくりと説明すると、データを入れ替えたり混合したりさせて暗号化させます。昔は AES-CBC という技術が主流でしたが、同じデータだと同じ暗号文になりパターンが決まり強度にかけるので、現代では AES-XTS という技術が用いられるようになりました。これは先述の AES-CBC と違い、同じデータでも位置が違えば違う暗号文になるので強度がより強くなりました。もちろん bitlocker だけでなく先述した web の暗号化のプロセスの一環に含まれても

います。その他もろもろありますが、代表例としてはこの 3 つが挙げられます。

2.4　bitlocker について

突然ですが皆さんはページ下のこのような画面に遭遇したことはないでしょうか。これは bitlocker といって、Windows のデータを保護するためのもので、この画面はその保護機構が働きユーザーに暗号化されたデータにアクセスするためにキーを入れることを要求している画面です。
初めてご存じになった方も、すでに見たことがあるよ、という方もこれがいきなり出てきたら驚くでしょう。というのも、この画面は本来は SSD（データを保存しているところ）などの部品に物理的な変更や、回復環境（Windows を修復するための環境）での機能を解放する際に出てくるはずですが、ひょんなことから（例えば普通にアップデートをした後）出てくることもあります。回復キーを使用者が控えている前提で出てくるので、もし回復キーがわからないとデータに二度とアクセスできなくなり、困ったことになります。ですのでこの記事を読んでくださった控えた覚えのないあなたは、設定から簡単にできるので今日にでも控えておくことをおすすめします。
次にこの bitlocker の仕組みについてですが、まずあなたがたユーザーが控えるのは VMK と呼ばれる、暗号化を解除するためのキーを保存しているものの暗号化を解除するキーになります（二重構造ってややこしい）。次に、私たちがキーを入力した後に VMK が解放されそれにより FVEK と呼ばれるもので暗号化されている本体のデータが復号化されます（この本体の暗号化に使われているのが AES）。ここで察しの良い方には気づいていただけたかと思いますが、これらのどちらかでも壊れると、bitlocker で暗号化されたデータにアクセスできなくなります。初期化するしかなくなりデータ救出は現実的ではなくなります。
ここで「bitlocker つけないほうが壊れた時のリスクとかを考えると安全じゃない？」と思った方、いるかもしれません。しかし bitlocker の暗号化を行わないことによるデメリットもあります。
一つは、管理者権限を乗っ取られるリスクがあることです。これは私も自分の PC で実際にやってみましたがいとも簡単に PC の主導権を奪取することができる、非常に脆弱な状態になっています。
次に、PC のデータを簡単に抜き取られることです。SSD などデータを保存している部品を抜き取り読み取れば平文がそのまま読み取れるわけですから、データは丸見えです。他にもたくさんありますが、これらはいかに PC が脆弱になるかを表しています。

図 2.1　bitlocker の画面

2.5　まとめ

暗号化というものは 3 文字で表せますが、その中にはたくさんの意味があります。種類もたくさんありますし、中には現代では使われなくなった技術もあります。ただ、今となってはデジタルのありとあらゆるところに「暗号化」が潜んでいます。友達と LINE するのにも、ネットでニュースを見るのにも、すべて暗号化がかかっています。それほどまでに暗号化という技術は私たちの生活に密接し、重要な役割を果たしています。
ところで、最近スーパーコンピューターというものをよく耳にしますが、いったいどれほどすごいのでしょうか。
世界一のエル・キャピタンと呼ばれるスーパーコンピューターは 1 秒に約 180 京回計算を行うことが可能ですが、AES-128（128bit の暗号化) を総当たり（1 から順に試すこと、ブルートフォース攻撃ともいう）しても、突破するまでに約 1 京年必要になります。なんとこれはうまくいった場合の理論値であり、実際はもっとかかるといわれています。どれほどすごいかというと、この 1 京年という数値は、宇宙の年齢の 100 万倍以上となっています。現代の技術ではまず解除不可能です。さらに AES-256 になると 256bit の暗号化となり、単純計算で 2 の 256-128 すなわち 2 の 128 乗倍の時間がかかることになります。つまり bitlocker などの暗号化は宇宙のどこかにあるかもしれないすごい技術を使えば突破できるかもしれませんが、今の地球を生きる私たちでは突破されることはまずないでしょう。
スーパーコンピューターなどの計算機構は将来どんどん進化していくでしょうが、それに合わせるように暗号化の技術も優れていくでしょう。
ここまで読んでくださった方、ありがとうございました。灘校文化祭をお楽しみください。

おまけ:bitlocker のイメージ

bitlocker についてイマイチわからなかった方へ。簡単に表すと私たちがもつ「鍵」は、玄関（データへの入り口へのたとえ）のドアの鍵が入っている「箱」の鍵で、私たちが入力したあとにその鍵をつかって玄関のドアをあけている、ようなものだと思っていただければ結構です。

第 3 章

Pandas で表を作ってみよう

82 回生 NF23

3.1　はじめに、Pandas について

Pandas とは、Python で使うことができる外部ライブラリの名前です。外部ライブラリというのは、Python の中の機能で、決まったコードを書くことによって使える便利な道具のようなものです。正直これを言っただけではわかりにくいので、早速使ってみましょう。

3.1.1　Pandas の導入

それでは、Pandas を使えるようにしていきます。Pandas は Python の中の機能なので、まずは Python を使える環境を準備します。まずはウェブ検索で「Google Colab」と調べて一番上に出てくるサイトをクリックしましょう。すると「ノートブックを開く」という枠が出てくると思います（出てこない場合は、Google アカウントでログインしてから入りなおしてみてください）。そうしたら、枠の右下に青い「ノートブックを新規作成」というボタンがあるので、クリックしましょう。下図のような画面が出て来たら成功です。そうしたら、Pandas を使えるようにするためのコードを書いていきます。画面に表示されている細長い長方形に下の

コードを書いて、Youtube の再生ボタンのようなボタンを押して下さい。これでコードを実行することができます。ちなみに、これからも違うコードを書いていきますが、新しいコードを書くときは左上のほうにある「+ コード」というボタンを押すことで書けるので覚えておいてください。

```
1 import pandas as pd
```

上のコードは、これから書くコードで「pd」と書けば Pandas の機能を使えるようにしています (インポートと呼ぶ)。Pandas を略して pd にしたという解釈で OK です。ちなみに略の仕方は pd でなくてもよいのですが、一般的なものはこれなので pd を使っています。

3.2 表を書く

ひとつ例として、直近三年の文化祭に関する表を作れるコードを書きました。

Listing 3.1 例

```
1 data = [[80,2026,'Polaris'],[79,2025,'weave'],[78,2024,'ODYSSEY',]]
2 df = pd.DataFrame(data , columns=['Number','Year','Name'])
3 print(df)
```

これを実行すると、下のような表ができるはずです。

この表には直近三年の文化祭が第何回で、何年に行われて、何という名前だったかという情報がかかれています。これからは、コードのどの部分が表を形づくっているか説明します。

3.2.1 'data' の中身

```
1 data = [[80,2026,'Polaris'],[79,2025,'weave'],[78,2024,'ODYSSEY',]]
```

この節では、上の一行を説明します。この一行では、表のデータ (名前や開催された年) の部分を 3 つの大かっこの記号の中に書いています。

```
1 [80,2026,'Polaris']
```

のような感じですね。「80」「2026」「Polaris」の 3 つのデータを「,」の記号によって分けています。この時、名前の部分のような文字を含むデータは、「'」の記号で囲って入力しないとエラーが出てしまうので気を付けましょう。ちなみに=の左側の「data」は、最初にやったインポートの作業と同じで=の右側に書いたデータを以降は「data」と書くだけで使えるようにするために使っています。

3.2.2 'df' の処理

```
1 df = pd.DataFrame(data , columns=['Number','Year','Name'])
```

この節ではこの行について解説します。この行では「data」から表に出力したいデータを取り出し、第何回、開催年、名前という形で分けています。それでは順楯列番に説明していきます。

```
1 pd.DataFrame
```

上の部分では、pd と書いて Pandas の DataFrame という機能を呼び出しています。DataFrame はデータを数 (縦列) と種類 (横列) のように 2 次元的に分ける機能で、表を作るときに使います。

```
1 data , columns = ['Number','Year','Name']
```

この部分では、上で触れた横列の中身を指定しています。表のデータである「data」を「Number」「Year」「Name」の三つに分けています。[80,2026,'Polaris'] というデータだったら、左から順に 80 が Number、2026 が Year、Polaris が Name になるわけですね。この横列のデータのことを「column」といいます。コードの中でも「columns」という単語が出ていますよね。ちなみに、対となる縦列はどこで設定しているのかというと、Pandas の方で勝手につけてくれています。これを「Index」といいます。Index は行が上から何番目かを示しているのですが、そのカウントは 0 からはじまり、例えば一行目は 0、二行目は 1 という形で表にかかれています。

3.2.3 print

```
1 print(df)
```

この節ではこの行を説明します。といっても説明することは少ないです。上の節で完成させた表のデータを print を使って出力するだけですね。これを忘れるとなにも出てこないので注意しましょう。

3.3 index や column を取り出す

上で取り扱った表は、index も column も三つだけでしたが、これがとても多くなると確認するのが大変です。そこで、表から特定の index や column を取り出す方法を書こうと思います。

3.3.1 index を取り出す方法

```
df [0:1]
df [0:3]
```

上のコードが、index を取り出せるコードです。このコードは、コロン (: の記号) の左側の index を最初の行として指定し、コロンの右側の index のひとつ上の行を終わりの行として指定します。1 行目のコードでは、index が 0 の行を始めとしてから、index が 1 の行のひとつ上の行、つまり 0 を終わりの行にするため、0 の行が出力されます。2 行目のコードだと、0 を始めとして 2 を終わりとするので、0 の行から 2 の行が出力されるというわけです。後述する column と違って、数字で行が決められているのでコードが短くて済みます。

3.3.2 column を取り出す方法

```
df['Year']
df['Number','Year','Name']
```

上のコードでは、column を取り出せます。index を指定するときとは違い、各 column の名前の間は「,」で区切ります。1 行目のコードでは、Year の列を出力することができます。また、2 行目のコードでは Number と Year と Name の列を出力できます。

3.3.3 index と column 両方を絞る

```
df.loc[0:0,'Year']
df.loc[0:2,['Year','Name']]
```

loc を用いることで、もっと細かく絞ることもできます。例えば一行目のコードでは、index が 0 の行で column が Year のデータを取り出すことができます。行や列が複数になっても同じことが可能です。ちなみに、index を指定するところと column を指定するところの間は「,」で区切られていますが、これは column 同士を区切る記号と同じであるため、コードを繋げて書くとエラーが出てしまいます。column を指定する場所は、2 行目でやっているように大かっこでくくりましょう。

3.4 おわりに

Pandas では、少しのコードを書くだけで表を簡単に生成することができます。今回は column が三つ、index が三つだけの表をつくりましたが、10 個、100 個と増やしていくことでもっと大きな表を作ることもでき、とても便利です。また、csv ファイルという形のファイルにすれば excel で作った表を Pandas で読み込むことだってできます。これを機会に、ぜひ Pandas や Python を初めてみるのはどうでしょうか。最後までお読みいただきありがとうございました。

第 4 章

Brainfuck の仕様やサンプルコード

82 回生 KitKat

4.1 はじめに

初めまして。82 回生の KitKat です。私は普段、競技プログラミングや exe の解析等をしています。
今回私が取り上げるのは、難解プログラミング言語の代表例である Brainfuck です。難解プログラミング言語とは、意図的に読み解くことが困難になるように作られたプログラミング言語のことで、Brainfuck のほかには、人間が書くことが不可能と言われている Malbolge や、二次元のグリッドに色を塗ることでプログラミングをする Piet などがあります。この記事では、Brainfuck の仕様の概要を紹介し、Web 版ではコードを生成するトランスパイラを自作します。

4.2 Brainfuck の仕様

4.2.1 言語仕様と本実装の設計方針

Brainfuck はコンパイラを極限まで小さくすることを目的として設計されました。命令は以下の 8 種類しかなく、それ以外の文字はすべてコメントとして無視されます。Brainfuck は処理系により細かな仕様が異なりますが、今回のトランスパイラや解説では、以下の仕様を採用します。

- メモリは右方向に十分長い 1 次元配列で、各セルは 0 で初期化される
- 各セルは上限・下限のない整数として扱い、オーバーフローは考慮しない

表 4.1　Brainfuck の命令一覧

命令	意味
>	ポインタをインクリメント（右のセルに移動）
<	ポインタをデクリメント（左のセルに移動）
+	ポインタが指す値をインクリメント（1 増やす）
-	ポインタが指す値をデクリメント（1 減らす）
.	ポインタが指す値を ASCII 文字として出力する
,	入力から 1 バイト読み込み、ポインタが指す値に代入する
[	ポインタが指す値が 0 なら対応する] の直後へジャンプする
]	ポインタが指す値が 0 でなければ対応する [へ戻る

4.2.2　実際のコードを見る

まず、Brainfuck で A（ASCII コード 65）を出力する単純なコードを見てみます。

```
++++++++[>++++++++<-]>+.
```

このコードが実行されたとき、メモリの状態は次のように遷移します（下線はポインタ位置）。

実行コード	メモリの状態	動作
（初期状態）	[0, 0, 0, ...]	
++++++++	[8, 0, 0, ...]	現在のセルを 8 にする
[	[8, 0, 0, ...]	値が非 0 なのでループに入る
>	[8, 0, 0, ...]	右のセルへ移動
++++++++	[8, 8, 0, ...]	右のセルに 8 を加算
<	[8, 8, 0, ...]	左のセルへ戻る
-	[7, 8, 0, ...]	カウンタを 1 減らす
...	...	左-1、右 +8 を左が 0 になるまで繰り返す
]	[0, 64, 0, ...]	左のセルが 0 になりループを抜ける。右のセルには $8 \times 8 = 64$ が残る
>	[0, 64, 0, ...]	右のセルへ移動する
+	[0, 65, 0, ...]	1 を足して 65 にする
.	[0, 65, 0, ...]	'A'（ASCII コード 65）を出力

このように、Brainfuck ではカウンタを減らしながら、別のセルに値を足し続けることで、乗算やデータのコピーを実現します。

4.2.3　手書きでの実装の限界

単純な文字出力であれば上記のように簡単に書けますが、少し複雑な処理を行おうとすると、コードの可読性は一気に低くなります。例として、フィボナッチ数列を計算し続けるプログラムを見てみましょう。

（Daniel B. Cristofani 氏[*1]）このプログラムは、メモリを 3 つのセルごとのブロックとして扱い、10 進数で演算を行っています。例として $5 + 8 = 13$ を計算する（繰り上がりが生じる）ケースを見てみましょう。全体の流れは以下の通りです。

1. 出力: 現在の項（数値）を ASCII 文字に変換して出力し、元の数値に戻す。
2. 計算: 次の項を計算するために数値を加算する。
3. 繰り上がり判定: 値が 10 以上の場合次の桁へ 1 を加算し、現在の桁を 1 の位だけにする。

- LF: 改行の ASCII コード（10）
- A: 数列の第 n 項（ここでは 5）
- T: 計算用のスペース兼マーカー（初期値 1）
- B: 数列の第 $n+1$ 項（ここでは 8）

表 4.2: フィボナッチ数列の計算

コード断片	直後のメモリ状態 （LF, A, T, B, A... の順）	動作の解説
初期化		
`>++++++++++` `>++++++>+>+++++++++`	`[10] [5] [1] [8]`	LF=10、A=5、T=1、B=8 をセット
`<`	`[10] [5] [1] [8]`	ポインタを T に戻す
出力		
`[+++++`	`[10] [5] [6] [8]`	T が 0 でない間ループする。まず T（$= 1$）に 5 を足して 6 にする
`[>+++++++++<-]`	`[10] [5] [0] [56]`	$T = 6$ 回ループで B に 8 を足す。合計 $6 \times 8 = 48$ が加算され、$B = 56$（ASCII '8'）となる
`>.<`	`[10] [5] [0] [56]`	'8' を出力する
`++++++`	`[10] [5] [6] [56]`	T を再度 6 にする
`[>---------<-]`	`[10] [5] [0] [8]`	6 回ループで B から 8 を引く。合計 $6 \times 8 = 48$ が減算され、$B = 8$ に戻る
`+<<<]`	`[10] [5] [1] [8]`	T を 1 に戻し、左の桁（今回は LF）へ移動する
計算フェーズ（$A \leftarrow B$, $T \leftarrow A + B$）		
`>.>>[`	`[10] [5] [1] [8]`	改行を出力し、右へ移動して計算を開始
`[-]`	`[10] [5] [0] [8]`	T を 0 にする
`<[>+<-]`	`[10] [0] [5] [8]`	A（$= 5$）を T に移す
`>>[<<+>+>-]`	`[10] [8] [13] [0]`	B（$= 8$）を A に移しつつ、T へ加算 $T = 5 + 8 = 13$ になる
繰り上がり判定 合計値 $T = 13$ を B に移すが、10 以上なので繰り上がり処理に入る		
`<[>+<-[>+<-[...`	`[10] [8] [4] [9]`	T から 9 回減算できるか試す $13 - 9 = 4$ なので、繰り上がり処理

[*1] `https://brainfuck.org/fib_explained.b`

コード断片	直後のメモリ状態	動作の解説
`>[-]`	`[10] [8] [4] [0]`	B に移した 9 を捨てて 0 にする
`>+>+`	`[10] [8] [4] [0] [1]`	隣のブロックの A と T に 1 を加算する（繰り上がりと計算継続フラグ）
`<<<-`	`[10] [8] [3] [0]`	現在の $T = 4$ からさらに 1 引き、一の位を 3 で確定する
`[>+<-]`	`[10] [8] [0] [3]`	残った T （$= 3$）を B へ移動する。これで一の位が 3 になる
`]]]]]]]]]]`	`[10] [8] [0] [3]`	ループを抜ける
`+>>>`	`[10] [8] [1] [3] [..]`	T を 1 に戻し、次の桁に移動する
`]<<<`	`[10] [8] [1] [3]`	メインループの先頭に戻る
結果		
（ループ終了）	`[10] [8] [1] [3] [1]..`	一の位は $B = 3$、次の桁に繰り上がり 1 が渡され、正しく 13 が計算できている

このように、Brainfuck でプログラミングを行う場合、計算ロジックが単純であっても、大量のポインタ操作などが必要になります。人間がこの複雑なメモリ操作を正確に管理し続けることは難しく、また最適化を怠り愚直に実装してしまうとコード量は人間が追えない長さになってしまいます。
これはサンプル版なのでここで終わりとなりますが、Web 版では、さらに Brainfuck のコードを自動生成するトランスパイラを実装します。ぜひ Web 版も読んでいただけると嬉しいです！

第 5 章

Siv3D でゲームを作ってみよう

82 回生 T.M

5.1 初めに

82 回生の T.M です。普段は Siv3D でゲーム開発をしています。今回は敵を上下左右のキーで移動して、敵をよけていくゲームを作っていきます。Siv3D のインストール方法に関しては URL を載せておきます。
URL : https://siv3d.github.io/ja-jp/

5.2 使う素材の設定

5.2.1 フォントと絵文字

```
1 Font font(40);
2 Font bigFont(80);
3 const Texture emoji1{ U"?"_emoji };
4 const Texture emoji2{ U"?"_emoji };
```

素材の設定ではまず、最初に設定するのは今回使うフォントの大きさと絵文字を設定します。フォントは大きさが 40 の小さいフォントと大きさが 80 の大きいフォントの二種類です。?の部分には自分の好きな絵文字を選んで入れることができ、emoji1 がプレイヤー、emoji2 が敵となっています。

5.2.2 ゲームの状況とプレイヤーの位置や大きさの設定

```
1 enum class GameState { Title, Countdown, Game, GameOver };
2 GameState state = GameState::Title;
3 Rect player(300, 400, 50, 50);
```

ゲームの状況を表す GameState(Title,Countdown,Game,GameOver) というクラスを作ります。2 行目で Title に変更しています。3 行目はプレイヤーの設定です。() の中には左から、x 座標,y 座標, 当たり判定の横幅, 当たり判定の縦幅となっています。

5.2.3 敵とスコアとストップウォッチの設定

```
1 const int enemyCount = 6;
2 int score = 0;
3 Array<Circle> balls;
4 Array<double> ballSpeeds;
5 for (int i = 0; i < enemyCount; ++i){
6         balls.push_back({ Random(0, Scene::Width()), Random(-500, 0), 25 });
7         ballSpeeds.push_back(Random(6.0, 8.0));
8 }
9 Stopwatch stopwatch(StartImmediately::No);
```

1 行目は enemyCount という一度に出る敵の数を設定する変数を作成しています。2 行目ではスコアを設定しています。3,4 行目で敵の当たり判定の大きさを設定する配列である Circle、敵の速さを設定する配列の ballspeeds を作っています。5 行目で enemyCount の数だけ繰り返し、6,7 行目で敵の当たり判定の位置や半径、敵の速さを決めています。9 行目で名前の通り stopwatch という名前のストップウォッチをストップさせています。

5.3 while 文での繰り返し操作

```
1 while (System::Update()){
```

while 文とは () の中が True の時に {} の中のコードが動きます。System::Update() は一生 (exit(終了)) を呼び出すまで true を返し続けるので、この While 文は一生 (Exit を呼び出すまで) 繰り返されます。Siv3D ではゲーム画面の右上の X ボタンを押すと自動的に exit が呼び出されるようになっていますので、ゲームを終了したいと思ったら右上の X ボタンを押せばいいです。

5.3.1 GameState=Title の時

```
1 if (state == GameState::Title){
2     font(U" 敵よけゲーム").drawAt(Scene::Center().x, 150);
3         Rect startButton(275, 300, 250, 100);
4         startButton.draw(Palette::Skyblue);
5         font(U" スタート").drawAt(startButton.center(),Palette::Black);
6         if (startButton.leftClicked()){
7                 state = GameState::Countdown;
8                 stopwatch.restart();
9         }
10 }
```

1 行目は、もし GameState=Title ならば、2 行目で敵除けゲームと画面に出力する。3,4 行目で startButton という長方形を設定し、描く。スタートという文字を startButton と同じ位置に出力する。6 行目は、もし startButton が左クリックされたなら、7,8 行目で、GameState を CountDown に変更して、stopwatch をリスタートさせています。

5.3.2 GameState=Countdown の時

```
1 if (state == GameState::Countdown) {
2         int count = 3 - (int)stopwatch.sF();
3         if (count > 0){
4                 bigFont(count).drawAt(Scene::Center());
5         }
6         else {
7                 state = GameState::Game;
8                 score = 0;
9                 stopwatch.reset();
10                for (int i = 0; i < enemyCount; ++i){
11                        balls[i].setPos(Random(0, Scene::Width()), Random(-500, 0));
12                }
13        }
14 }
```

1 行目は、もし GameState=Countdown であるならば 2 行目整数変数 count を 3 - (stopwatch を小数化して小数点以下を切り捨てたもの (これは整数である)) とします。3 行目は、もし count ¿ 0 ならば 4 行目大きいフォントで count をゲーム画面の真ん中に表示します。そうでなければ、7,8,9 行目 GameState を Game に変更して、スコアを 0 にし、ストップウォッチをリセットします。10 行目で enemyCount だけ繰り返します。11 行目で敵をランダムな位置に移動させます。

5.3.3 GameState = Game の時

```
1 if (state == GameState::Game){
2         if (KeyLeft.pressed()) player.x -= 7;
3         if (KeyRight.pressed()) player.x += 7;
4         if (KeyUp.pressed()) player.y -= 5;
5         if (KeyDown.pressed()) player.y += 5;
6         player.x = Clamp(player.x, 0, Scene::Width() - player.w);
7         player.y = Clamp(player.y, 0, Scene::Height() - player.h);
8         for (int i = 0; i < enemyCount; ++i) {
9                 balls[i].y += ballSpeeds[i];
10                if (balls[i].y > Scene::Height()) {
11                        balls[i].y = 0;
12                        balls[i].x = Random(0, Scene::Width());
13                        score++;
14                }
15                if (player.intersects(balls[i])) {
16                        state = GameState::GameOver;
17                }
18                emoji2.scaled(0.5).drawAt(balls[i].x, balls[i].y);
19        }
```

```
20          emoji1.scaled(0.5).drawAt(player.center());
21          font(Uスコア": {}"_fmt(score)).draw(20, 20);
22  }
```

1 行目は (もし GameState = Game であるならば) です。2,3,4,5 行目はそれぞれ ← キーが押されたとき player の x 座標を-7、→ キーが押されたとき player の座標を +7、↑ キーが押されたとき player の y 座標を-5、↓ キーが押されたとき player の座標を-7 します。数学のグラフなどを書くときの原点からの位置の x 座標,y 座標の感覚とずれていると思いますが、Siv3D では X,Y 座標はゲームスクリーンの左上を原点としているので、↓ キーを押すことで player の y 座標が増えるようにしています。6,7 行目の clamp は指定した値をその範囲内に収めるようにする関数です。この場合では、player がゲーム画面の端に当たった時にそれ以上動けないようにするために使っています。8 行目で enemyCount の数だけ繰り返します。9 行目は i 番目の敵の y 座標を i 番目の ballSpeeds ずつ増やします。これで敵が ↓ に移動していきます。10 行目,11 行目,12 行目,13 行目はもし敵の y 座標がゲーム画面の y 座標を超えたならば、敵の y 座標を 0 に戻し、x 座標をランダムな位置にして、score を +1 します。15 行目,16 行目はもし player が敵に触れたならば、GameState を GameOver にします。18 行目,20 行目で emoji2(敵),emoji1(player) を描きます。

5.3.4 GameState = GameOver の時

```
1  if (state == GameState::GameOver) {
2          bigFont(U" ゲームオーバー").drawAt(Scene::Center().x, 200);
3          font(U" スコア: {}"_fmt(score)).drawAt(Scene::Center().x, 300);
4          font(U" エンターでタイトルへ").drawAt(Scene::Center().x, 400);
5          if (KeyEnter.down()) {
6                  state = GameState::Title;
7                  player.setPos(300, 400);
8          }
9  }
```

1 行目は (もし GameState = GameOver であるならば、) です。2,3,4 行目で大きいフォントでゲームオーバー、小さいフォントでスコア、小さいフォントでエンターでタイトルへと描きます。5.6,7 行目でもしエンターキーが押されたならば、GameState を Title に変更して、player の x 座標を 300,y 座標を 400 にします。これでコードは以上です。今までのすべてのコードを統合して一つのコードにするときはインデントに気を付けて統合してください。

5.4 終わりに

```
if (state == GameState::Game){
        if (KeyLeft.pressed()) player.x -= 7;
        if (KeyRight.pressed()) player.x += 7;
        if (KeyUp.pressed()) player.y -= 5;
        if (KeyDown.pressed()) player.y += 5;
        player.x = Clamp(player.x, 0, Scene::Width() - player.w);
        player.y = Clamp(player.y, 0, Scene::Height() - player.h);
        for (int i = 0; i < enemyCount; ++i){
                balls[i].y += ballSpeeds[i];
                if (KeySpace.pressed() && KeyB.pressed()){
                        balls[i].y = -10;
                        balls[i].x = Random(0, Scene::Width());
            score++;
                }
                if (balls[i].y > Scene::Height()){
                        balls[i].y = 0;
                        balls[i].x = Random(0, Scene::Width());
                        score++;
                }
                if (player.intersects(balls[i])){
                        state = GameState::GameOver;
                }
                emoji2.scaled(0.5).drawAt(balls[i].x, balls[i].y);
        }
        emoji1.scaled(0.5).drawAt(player.center());
        font(U" スコア: {}"_fmt(score)).draw(20, 20);
}
```

section3.3 のコードを少し改変してみました。スペースキーと B キーを同時に押してみてください。Siv3D ではコードを少し変更するだけでチートを付け加えることもできます。ぜひ僕の作ったゲームコードを変更したり、新しいゲームを作ったりして遊んでみてください。

第 6 章

React2Shell（CVE-2025-55182）の検証

81 回生 mikan0131

6.1 はじめに

こんにちは、81 回生の mikan0131 といいます。普段は Web アプリの個人開発などをしているのですが、今回は Web アプリケーションフレームワーク React で最近見つかった致命的な脆弱性「React2Shell」を解説し、実際に攻撃してみようと思います。
セキュリティに興味のある人もない人も、最後まで楽しんでもらえたら嬉しいです。そして、サイバー攻撃が他人ごとではないことを体感してもらえたらと思います。

注意

本述では、実際に使えるハッキングの手口を紹介しています。
内容を実際に運営されている Web サービスで試すのはくれぐれもご遠慮ください。今回は自前で構築した模擬的な Web アプリに攻撃を行います。

6.2 React2Shell とは

React2Shell は 2025 年 12 月 3 日に公開された React または Next.js[*1] に関する致命的な脆弱性です。正式名称は CVE-2025-55182 で、CVSS は 10.0 を記録しました。CVSS は最大値が 10.0 なので、これ以上危険な脆弱性は理論上存在しないということです。React2Shell が公開されると同時に公式のセキュリティパッチも公開されましたが、公開後、React2Shell を使った大規模な攻撃、複数の Web サービスのクラッシュが確認され、IT 業界では一時期この脆弱性がかなり問題視されました。
この脆弱性を使えば、Web アプリを運営しているサーバー内の権限をほとんど掌握することができます。ファイル書き込むのもよし、プログラムを実行するのもよし、さらに言えば、パスワードを変えてサーバー管理者を締め出すこともできるかもしれません。
今回は、そんな React2Shell の仕組みを解説し、模擬的な攻撃を仕掛けてみようと思います。

[*1] 現在多くの Web アプリに利用されているフレームワークの一つ。TikTok や Notion、OpenAI もこのフレームワークを利用しています。

6.3　知っておきたい事前知識

では、今から React2Shell の仕組みを解説していくのですが、その前に、React2Shell にかかわるいくつかの重要な知識について解説します。

6.3.1　React Flight

React2Shell の脆弱性の肝となったのが、React Flight プロトコル[*2]という技術です。これは、React や Next.js が裏で動かしている通信方式で、より高速に、効率よく通信を行うために導入されました。導入されたのは React18 からです。
React Flight はコンポーネント[*3]から Promise 型の非同期関数[*4]まで、様々なものを効率よく通信することができます。React Flight では、まずサーバーがクライアントに送信したいコンテンツをある特定のルールに従って変換します。そして、変換されたデータをクライアントに送り、クライアントサイドでデータを復元し、Web サイトを構築することができます。

図 6.1　React Flight の仕組み

6.3.2　Server Component

現在、Next.js には二種類のコンポーネントが存在します。それらを Server Component と Client Component と呼びます。Client Component は、クライアントにソースコードを送信して、クライアント側で Web サイトを構築するコンポーネントですが、Server Component はサーバーサイドでコンポーネントを構築したうえで、クライアントに送信します。このとき、クライアントにコンポーネントのデータを送信する際、React Flight が利用されます。
また、Server Component に特徴的なのは、Server Actions の存在です。Server Actions は、クライアントからデータをサーバーに渡し、サーバーでの処理を経てクライアントサイドの Web サイトにコンテンツを反映する仕組みのことです。
Server Actions はサーバーサイドで処理が実行されるため、セキュリティ面でもパフォーマンス面でも様々なメリットがあります。しかし、Server Actions ではクライアント側から与えられた値を処理するので、不正な値を使用した不正なプログラムが実行されてしまうかもしれないというリスクもあります。

[*2] 通信をするとき、正しく情報を伝達できるように泡勝目決めて置く通信方式のルールのこと
[*3] Web サイトを構成する部品のようなもの
[*4] API リクエストなど、実行されるまでに時間のかかる関数のこと

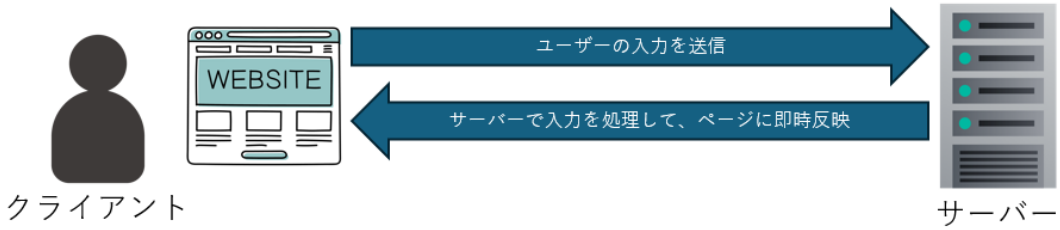


図 6.2　Server Actions の仕組み

6.4　React2Shell で利用された脆弱性

それでは、事前知識を頭に入れてもらったところで、React2Shell で利用された脆弱性について解説します。React2Shell は、先ほど解説した React Flight と Server Component の Server Actions を掛け合わせたときに発生する脆弱性を利用したものです。
ここからは、React2Shell が具体的にどのような脆弱性をついたものであるかを説明します。

6.4.1　React Flight の例外

React Flight はサーバーからクライアントへコンテンツを送信するときに使われると述べました。しかし、これには例外があります。それは Server Actions を利用したときです。実は、Server Actions の通信方式にも React Flight が使われています。これは何を意味するのでしょうか。それは、React Flight の変換プログラムには通常サーバー側からのデータ、つまり信頼性の高いデータしか渡されないのに対し、Server Actions を通じてクライアントからの入力、つまり信頼性の低いデータを渡される可能性があるということです。
このことによって、ユーザーからの入力により、サーバーの脆弱性を突ける可能性が高まると思われます。

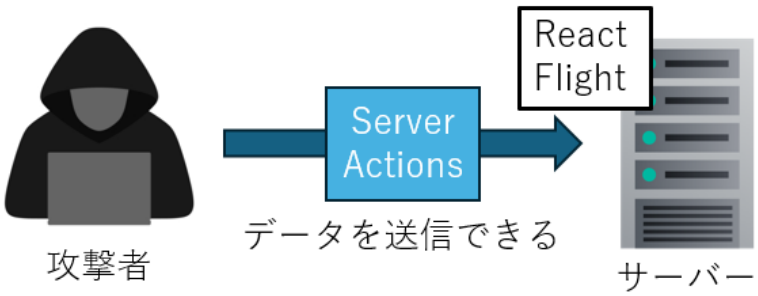


図 6.3　Server Actions による React Flight へのデータ送信

6.4.2　React Flight のバグ

最も重要な脆弱性は、React Flight の変換・復元プログラムのバグ[*5]です。今回の React Flight のバグは何だったのでしょうか？
それは、簡単に言うと、「React Flight が変換するデータのことを信用しすぎている」ということです。どういうことかというと、例えば、ある攻撃者から、React Flight 変換プログラムに不正な値が与えられたとしましょう。この時、一般的には、そういった不正な値をはじくプログラムも変換プログラムの中に実装されていて、それによって不正なプログラムが実行されないようになっています。しかし、React Flight の変換プログラムにはそのような条件分岐が実装されていません。そのため、攻撃者からの不正な入力で攻撃者の意図するプログラムがサーバー内で実行されてしまう恐れがあります。

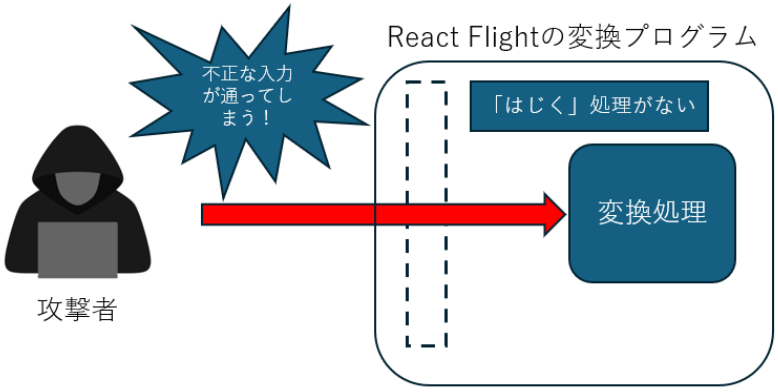


図 6.4　React Flight 変換プログラムの脆弱性

6.5　React2Shell の手口

React2Shell を使えばこのような脆弱性を突いて、サーバー内で意図した不正なプログラムを実行・そしてサーバー内の権限を獲得することができます。具体的には、まず、Server Actions がサーバーに送信している入力のように見せかけた偽データを Server Actions を通じてサーバーに送信します。そして、そのデータがサーバー内の React Flight の変換プログラムで変換される過程で、攻撃者の意図するコマンドをサーバー内で実行できるようにすることができます。
実際にこの攻撃方法を使うと、サーバーの環境変数の奪取・ファイルの作成・プログラム実行など、本当に様々なことをサーバー内で行うことができます。
これはサンプル版なのでここで終わりですが、Web 版ではさらに React Flight のプログラムの仕組みをさらに掘り下げて、どんな仕組みで不正なプログラムが実行されるのかを解説します。
ここまで読んでくださり、ありがとうございました！ぜひ Web 版部誌もお楽しみください！

[*5] ここでいう「バグ」は、攻撃者が利用することのできるプログラム上のミスという意味です。

第 7 章

API の結果を Discord に送信する bot の開発

79 回生 山下航平

7.1 はじめに

79 回生・高校 3 年の山下航平です。昨年はパソコン部の部長を務めていました。

普段は競技プログラミングに取り組んでいますが、最近は塾の都合であまり参加できていません。大学受験を意識し始めている今日この頃です。

7.2 Discord のボット

Google Apps Script（以下、GAS）を使うと、簡単に Discord のボットを作ることができます。

GAS のコードはほとんどが JavaScript に近いため、JavaScript を書ける人にとっては手軽に扱えると思います。

7.2.1 Discord と GAS を連携する方法

1.Discord の開発者モードを有効にする 2. メッセージ投稿したいチャンネルのウェブフックを作成する 3.Discord にメッセージを投稿するプログラムを作成する

7.3 ボットのきっかけ

今回ボットを作ろうと思ったきっかけは、とあるイベントに空きが出たかどうかを自動で確認し、空きが出たら通知してくれる仕組みを作りたかったからです。そのイベントの空席情報の取得方法を調べている中で API を見つけたため、今回はそこからデータを取得することにしました。

7.4 ボットの仕組み

それでは、ボットの仕組みについて説明します。API では、都道府県やリーグなどの条件を指定して検索することができます。今回は、都道府県を「兵庫・大阪・奈良・京都」の 4 つ、リーグを「オープン・シニア」の 2 つとし、合計 8 通りの条件で検索を行います。API の形式の例は、以下のようになっています。

Listing 7.1　API

```
{
  ”event_date_params”: ”20260408”,
  ”event_date”: ”04/08”,
  ”event_date_week”: 水””,
  ”event_started_at”: ”12:30”,
  ”event_ended_at”: ””,
  ”id”: 1061,
  ”date_id”: 1752680,
  ”shop_id”: 11037,
  ”prefecture_name”: 兵庫県””,
  ”deck_count”: 枚”60”,
  ”zip_code”: 6008211,
  ”venue”: ””,
  ”event_holding_id”: 953007,
  ”event_type”: 2,
  ”csp_flg”: 1,
  ”event_league”: 1114,
  ”regulation”: スタンダード””,
  ”entry_fee”: 円”500”,
  ”capacity”: 64,
  ”shop_name”: ”店 XXXXX”,
  ”leagueName”: オープン””,
  ”entryStatus”: 満席””,
  ”entryStatusCode”: 6
}
```

ここで、*entryStatusCode* が 6 の時に満席、2 の時に空席であることがわかったため、

”entryStatusCode”: 2 の文字列を数えることで、空席のイベントの数を検知することができました。イベントの空き状況の変化を検知するには、前回の実行結果を保存しておく必要があります。そこで、Google スプレッドシート（以下、スプシ）と GAS を連携させ、スプシに保存された値との差分を確認しながら更新する仕組みにしました。

7.5　コードの解説

コードの簡単な説明を載せておきます。(画像内の一部情報は伏せています)。

Listing 7.2　プログラム 1

```
function getData(cell) {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  Logger.log(cell);
  var data = sheet.getRange(cell).getValue();
  return data;
}

function writeData(content, cell) {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  sheet.getRange(cell).setValue(content);
}

function output(content) {
  const message = {
    ”content”: content,
```

```
16      "tts": false
17    }
18    const discordWebHookURL = "https://discord.com/api/webhooks/1481584181564805271/
          JauqeFJXPe3XZpBA2SruLZNZ2phJFHfBDVyx7mGASUziJA7DDl3E05ctdmSAZi35sDsX";
19    const discordWebHookURL_test = "https://discord.com/api/webhooks/1458470666268577792/
          OEnw4khsroI0tITv2btsQX5DSuzGoPgqL27pWBOpAW0DpzavmIMCA0BN070lABz6XV1r";
20    const param = {
21      "method": "POST",
22      "headers": { 'Content-type': "application/json" },
23      "payload": JSON.stringify(message)
24    }
25    Logger.log(content);
26    UrlFetchApp.fetch(discordWebHookURL, param);
27  }
```

最初に、getData、writeData、output の 3 つの関数を定義しています。

getData は、スプシの指定した位置の値を取得する関数です。

writeData は、指定した位置に値を書き込む関数です。

output は、Discord のボットにメッセージを送信する関数です。

Listing 7.3　プログラム 2

```
1  function do_main() {
2    try {
3      myFunction();
4    } catch (e) {
5      const message = キャンセル待ち‘でエラー発生中。botエラー内容
6  : ${e.stack}‘;
7      Logger.log(e);
8      output(message);
9      throw new Error(message);
10   }
11 }
```

次に、do_main 関数では、GAS のトリガーという機能を使うことで 5 分ごとに実行する設定を行っています。内部で myFunction 関数を呼び出し、エラーが発生した場合にはその内容を Discord に送信します。

Listing 7.4　プログラム 3

```
1  function myFunction() {
2    prefect = ["36", "37", "39", "38"];
3    prefect2 = 兵庫 ["", 大阪"", 奈良"", 京都""];
4    league_type = ["1", "3"];
5    league_type2 = オープン ["", シニア""];
6    cell_num = ["A1", "A2", "A3", "A4", "B1", "B2", "B3", "B4"];
7    var message = "<@000000000000000000>\n";
8    let check = 0;
9
10   for (let i = 0; i < 4; i++) {
11     for (let j = 0; j < 2; j++) {
12       var url2 = "https://xxxxxxx.xxxxxxxx.com/event_search?prefecture[]=" + prefect[i] + "&
             league_type[]=" + league_type[j] ;
13       Logger.log(url2);
14       var response2 = UrlFetchApp.fetch(url2, {
15         muteHttpExceptions: true
16       });
17       var status = response2.getResponseCode();
```

```
18        if (status == 404) {
19          var count = 0;
20          Logger.log(i + j * 4);
21          var origin = getData(cell_num[i + j * 4]);
22          if (count != origin) {
23            check = 1;
24            var chang = count - origin;
25            writeData(count, cell_num[i + j * 4]);
26            message += 先着受付中' の' + league_type2[j] + ' ' + prefect2[i] + ' の数は
                  ' + count + ' (' + (chang > 0 ? '+' : '') + chang + ') 件です。\n';
27          }
28          else {
29            message += 先着受付中' の' + league_type2[j] + ' ' + prefect2[i] + ' の数は' + count + ' 件で
                  す。\n';
30          }
31          Logger.log先着受付中 ("" の数: ' + i + ' ' + j + ' ' + count);
32          continue;
33        }
34        var text = response2.getContentText();
35        Logger.log(text);
36        var count = (text.match(/"entryStatusCode":2/g) || []).length;
37        Logger.log(i + j * 4);
38        var origin = getData(cell_num[i + j * 4]);
39        if (count != origin) {
40          check = 1;
41          var chang = count - origin;
42          writeData(count, cell_num[i + j * 4]);
43          message += 先着受付中' の' + league_type2[j] + ' ' + prefect2[i] + ' の数は
                ' + count + ' (' + (chang > 0 ? '+' : '') + chang + ') 件です。\n';
44        }
45        else {
46          message += 先着受付中' の' + league_type2[j] + ' ' + prefect2[i] + ' の数は' + count + ' 件です。
                \n';
47        }
48        Logger.log先着受付中 ("" の数: ' + i + ' ' + j + ' ' + count);
49      }
50    }
51    if (check == 1) {
52      output(message);
53    }
54  }
```

最後に、myFunction 関数では、イベントの空き状況を確認し、その結果を Discord に送信します。

8 通りの条件を for 文で順に処理し、前回の結果（5 分前）と異なる場合、つまりイベントに空きが出た、または埋まった場合にのみ通知を行います。

7.6 おわりに

このボットを作ったことで、イベントの空き状況を毎回手動で確認する必要がなくなり、とても便利になりました。ぜひ皆さんも Discord ボットの作成に挑戦してみてください。

第 8 章

Android のアプリを作ろう

81 回生 ぱんだ

8.1　はじめに

こんにちは。81 回生のぱんだです。この記事では、自分で Android のアプリを作っていこうと思います。

8.2　環境

Android Studio Panda 2 — 2025.3.2 [*1]

デバイスはバーチャル・現実ともに Pixel 6a を使用しています。（ものによって動作や画面サイズが違います。今回載せている写真はすべて Pixel 6a のものです。）

8.3　コード

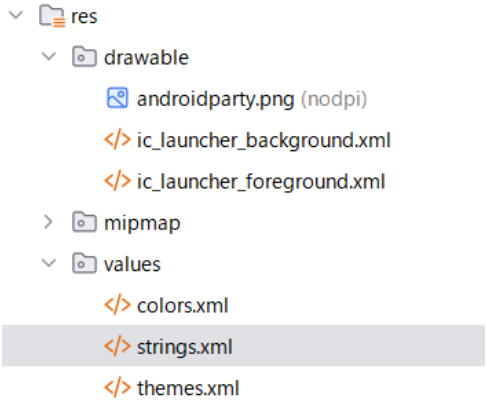
今回作るのはこのようなアプリです。（アプリを開くと以下の写真のような画面が出るだけのもの。）

コードはいくつかの部分に分かれているので、順番に説明していきます。

[*1] Android Studio とは、「Android アプリ開発用の公式の統合開発環境」（https://developer.android.com/studio/intro?hl=ja より引用）です。Android でアプリを作るならこれが一番いいのではないかと思います。

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            HappyBirthdayTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    GreetingImage(
                        message = stringResource( id = R.string.happy_birthday_text),
                        from = stringResource( id = R.string.signature_text),
                        modifier = Modifier.padding( all = 8.dp)
                    )
                }
            }
        }
    }
}
```

まずこの部分では、全体の設定をしています。Surface() 内では、modifier によって、「最大サイズの文字の表示をする」こと、color によって「背景に合わせて色を変える」ことの設定をしています。（この colorScheme の機能は Android12 以降でしか使用できないそうです。）GreetingImage() 関数で、全ての描画を行っています（背景画像、表示文字)。ここでこの関数の message,from に渡されている”R.string. ○○”とはなんでしょうか。これは、コード内に直接表示する文章を書き込むのではなく、外部に置いてあるファイルから読み取ることで、表示する文字を変えやすくしています。



ここの strings.xml 内で以下のように設定しているため、R.string.happ_birthday_text を指定すると「Happy Birthday [name]!」を取得できるのです。

¡string name=”happy_birthday_text”¿Happy Birthday [name]!¡/string¿

GreetingImage() 関数の内容については後述します。

```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Column(
        verticalArrangement = Arrangement.Center,
        modifier = modifier

    ) {
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp,
            textAlign = TextAlign.Center
        )
        Text(
            text = from,
            fontSize = 36.sp,
            modifier = Modifier
                .padding( all = 16.dp)
                .align(alignment = Alignment.CenterHorizontally)
        )
    }
}
```

次にここでは、文字を表示する設定をしています。ここで GreetingText() 関数を作成します。Column() 内では、この後の Text 要素 2 つを真ん中揃えで配置することを設定しており、その後の Text 要素では「表示するテキストの内容」（ここでは、GreetingText に渡された message,from を指定している）やフォントのサイズなどを指定しています。

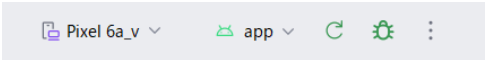
```
@Composable
fun GreetingImage(message: String, from: String, modifier: Modifier = Modifier) {
    val image = painterResource( id = R.drawable.androidparty)
    Box(modifier){
        Image(
            painter = image,
            contentDescription = null,
            contentScale = ContentScale.Crop,
            alpha = 0.5F
        )
        GreetingText(
            message = message,
            from = from,
            modifier = Modifier
                .fillMaxSize()
                .padding( all = 8.dp)
        )
    }
}
```

この項では、背景画像の設定を行っています。ここで GreetingImage() 関数を設定しています。まず、image で背景画像の設定をしています。先ほどの R.string の部分の解説と同じようなことですね。今度は drawable の方から背景画像の png を取得しています。これを後の Image() に渡し、背景に配置しています。GreetingText() 内では、GreetingImage() に渡された message と from がそのまま渡されていますね。最初の部分で R.string が渡されているため、これがここの GreetingText() に渡されることになります。これによって外部ファイル内の文字列を指定したフォントサイズで描画することが出来ています。

これらによって最初の画像のようなアプリが完成するというわけですね。

8.4 動作確認

PC 上で動作するか試してみましょう。右側の Device Manager を開きます。（既にバーチャルデバイスを設定している方は読み飛ばしていただいて結構です。）Add a new device → Create virtual device を押し、好きな機種を選択しダウンロードします。（実機で動かしたい方は、持っているデバイスと同じものを選ぶといいでしょう。）

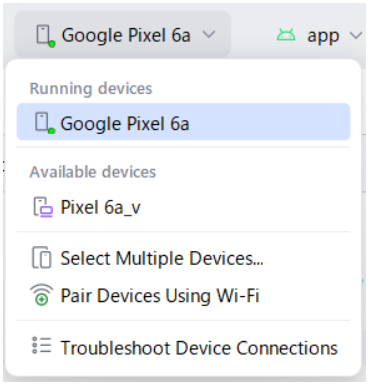


左側の部分でデバイス選択をし、右の run を押すとエミュレーターが作動しアプリが開きます。

それでは、実機で動かしてみましょう。方法はいくつかありますが、今回はその中の 2 つを紹介します。1 つ目は、apk ファイルを使用する方法です。メニューの Build → Generate Signed Bundle/APK を選択し、出てきた画面で APK を選択し次へ進み、Key store path の下の Create new を選択します。

Key store path で保存場所を設定し、Password を 2 か所それぞれと First and Last name を設定します。次の画面で release を選択し create を押すと、ビルドが始まり `C:\Users\〇〇\AndroidStudioProjects\プロジェクト名\app\release` に apk ファイルが生成されます。これを任意の端末に入れて実行すればその端末にアプリを入れることが出来ます。

２つ目は、PC とデバイスを直接接続する方法です。デバイスの設定で「USB デバッグ」を有効にした状態で PC とデバイスを有線で接続します。



接続したデバイスを選択した状態で run を実行すると、自動的に接続しているデバイスでそのアプリが立ち上がります。

8.5 おわりに

今回は文字を表示するだけのアプリを制作しましたが、もちろん他にコードを書けばゲームなどを作ることが出来ます。皆さんもやってみてください。

参考

compose を用いた Android アプリ開発の基礎
`https://developer.android.com/courses/android-basics-compose/unit-1?hl=ja`